RDF 1.2 Concepts and Abstract Syntax

W3C Editor's Draft 16 January 2025

▼ **More details about this document**

This version:
https://w3c.github.io/rdf-concepts/spec/

Latest published version:
https://www.w3.org/TR/rdf12-concepts/

Latest editor's draft:
https://w3c.github.io/rdf-concepts/spec/

History:
https://www.w3.org/standards/history/rdf12-concepts/
Commit history

Latest Recommendation:
https://www.w3.org/TR/rdf11-concepts

Editors:
Olaf Hartig
Pierre-Antoine Champin
Gregg Kellogg
Andy Seaborne

Former editors:
Richard Cyganiak
David Wood (Chair)
Markus Lanthaler
Graham Klyne
Jeremy J. Carroll
Brian McBride

Feedback:
GitHub w3c/rdf-concepts (pull requests, new issue, open issues)
public-rdf-star-wg@w3.org with subject line `[rdf12-concepts] … message topic …` (archives)

Abstract

The Resource Description Framework (RDF) is a framework for representing information in the Web. This document defines an abstract syntax (a data model) which serves to link all RDF-based languages and specifications. The abstract syntax has two key data structures:

- RDF graphs are sets of subject-predicate-object triples, where the elements may be IRIs, blank nodes, datatyped literals, or triple terms. They are used to express descriptions of resources.
- RDF datasets are used to organize collections of RDF graphs, are comprised of a default graph and zero or more named graphs.

RDF 1.2 introduces triple terms as another kind of RDF term which can be used as the object of another triple. RDF 1.2 also introduces directional language-tagged strings, which contain a base direction element that allows the initial text direction to be specified for presentation by a user agent.

RDF 1.2 Concepts introduces key concepts and terminology for RDF 1.2, discusses datatyping, and the handling of fragment identifiers in IRIs within RDF graphs.

Status of This Document

This section describes the status of this document at the time of its publication. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at https://www.w3.org/TR/.

This document is part of the RDF 1.2 document suite. It is the central RDF 1.2 specification and defines the core RDF concepts. Test suites and implementation reports of a number of RDF 1.2 specifications that build on this document are available through the *RDF 1.1 Test Cases* document [RDF11-TESTCASES].

RDF 1.2 Concepts is an update to [RDF11-CONCEPTS], which was itself, an update to [RDF-CONCEPTS-20040210].

> EDITOR'S NOTE
>
> Determine how to reference 1.2 test cases.

This document was published by the RDF-star Working Group as an Editor's Draft.

Publication as an Editor's Draft does not imply endorsement by W3C and its Members.

This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress. Future updates to this specification may incorporate new features.

This document was produced by a group operating under the W3C Patent Policy. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

This document is governed by the 03 November 2023 W3C Process Document.

§ Set of Documents

This document is one of ten RDF 1.2 and twelve SPARQL 1.2 documents produced by the RDF-star Working Group.

▶ **List of documents**

Table of Contents

1. Introduction

This section is non-normative.

The *Resource Description Framework* (RDF) is a framework for representing information in the Web.

This document defines an abstract syntax (a data model) which serves to link all RDF-based languages and specifications, including the following:

- the formal model-theoretic semantics for RDF [RDF12-SEMANTICS]
- serialization syntaxes for storing and exchanging RDF such as *RDF 1.2 Turtle* [RDF12-TURTLE] and *JSON-LD 1.1* [JSON-LD11]
- the *SPARQL 1.2 Query Language* [SPARQL12-QUERY]
- the *RDF 1.2 Schema* [RDF12-SCHEMA]

1.1 Graph-based Data Model

The core structure of the abstract syntax is a set of triples, each consisting of a subject, a predicate and an object. A set of such triples is called an RDF graph. An RDF graph can be visualized as a node and directed-arc diagram, in which each triple is represented as a node-arc-node link.

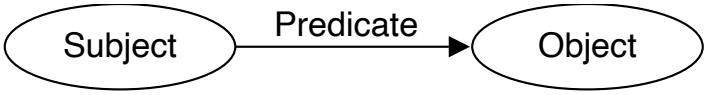

Figure 1 An RDF graph with two nodes (Subject and Object) and a triple connecting them (Predicate)

There can be four kinds of nodes in an RDF graph: IRIs, literals, blank nodes, and triple terms.

§ 1.2 Resources and Statements

or triple terms

Any IRI or literal ***denotes*** something in the world (the "universe of discourse"). These things are called ***resources***. Anything can be a resource, including physical things, documents, abstract concepts, numbers and strings; the term is synonymous with "entity" as it is used in *RDF 1.2 Semantics* [RDF12-SEMANTICS]. The resource denoted by an IRI is called its referent, and the resource denoted by a literal is called its literal value. Literals have datatypes that define the range of possible values, such as strings, numbers, and dates. Special kinds of literals — language-tagged strings and directional language-tagged strings — respectively denote plain-text strings in a natural language, and plain-text strings in a natural language including an initial text direction.

Asserting an RDF triple says that *some relationship, indicated by the predicate, holds between the resources denoted by the subject and object*. This statement corresponding to an RDF triple is known as an ***RDF statement***. The predicate itself is an IRI and denotes a ***property***, that is, a resource that can be thought of as a binary relation. ~~(Relations that involve more than two entities can only be indirectly expressed in RDF [SWBP-N-ARYRELATIONS].)~~

and triple terms

Unlike IRIs and literals, blank nodes do not identify specific resources. Statements involving blank nodes say that something with the given relationships exists, without explicitly naming it.

The proposition denoted by a triple term holds true in a distinct world than the current universe of discourse, where the statements in the RDF graph are meant to be true; this is why we say the a triple term is not asserted.

the denotation of

The proposition holds

A triple term denotes a ***proposition***. This is a logical, abstract resource, identified by its constituent predicate property and subject and object resources. ~~It is simply~~ true in the universe only if a corresponding triple with the same subject object and predicate is asserted. ~~In this way, an RDF statement makes its corresponding, abstract proposition true.~~

§ 1.3 The Referent of an IRI

The resource denoted by an IRI is also called its ***referent***. For some IRIs with particular meanings, such as those identifying XSD datatypes, the referent is fixed by this specification. For all other IRIs, what exactly is denoted by any given IRI is not defined by this specification. Other specifications may fix IRI referents, or apply other constraints on what may be the referent of any IRI.

Guidelines for determining the referent of an IRI are provided in other documents, like *Architecture of the World Wide Web, Volume One* [WEBARCH] and *Cool URIs for the Semantic Web* [COOLURIS]. A very brief, informal, and partial account follows:

- By design, IRIs have global scope. Thus, two different appearances of an IRI denote the same resource. Violating this principle constitutes an IRI collision [WEBARCH].
- By social convention, the IRI owner [WEBARCH] gets to say what the intended (or usual) referent of an IRI is. Applications and users need not abide by this intended denotation, but there may be a loss of interoperability with other applications and users if they do not do so.
- The IRI owner can establish the intended referent by means of a specification or other document that explains what is denoted. For example, the *The Organization Ontology* [VOCAB-ORG] specifies the intended referents of various IRIs that start with `http://www.w3.org/ns/org#`.
- A good way of communicating the intended referent is to set up the IRI so that it dereferences [WEBARCH] to such a document.
- Such a document can, in fact, be an RDF document that describes the denoted resource by means of RDF statements.

Perhaps the most important characteristic of IRIs in web architecture is that they can be dereferenced, and hence serve as starting points for interactions with a remote server. This specification is not concerned with such interactions. It does not define an interaction model. It only treats IRIs as globally unique identifiers in a graph data model that describes

resources. However, those interactions are critical to the concept of *Linked Data Design Issues*, [LINKED-DATA], which makes use of the RDF data model and serialization formats.

§ 1.4 RDF Vocabularies and Namespace IRIs

An ***RDF vocabulary*** is a collection of IRIs intended for use in RDF graphs. For example, the IRIs documented in [RDF12-SCHEMA] are the RDF Schema vocabulary. RDF Schema can itself be used to define and document additional RDF vocabularies. Some such vocabularies are mentioned in the Primer [RDF12-PRIMER].

The IRIs in an RDF vocabulary often begin with a common substring known as a ***namespace IRI***. Some namespace IRIs are associated by convention with a short name known as a ***namespace prefix***. Some examples:

Some example namespace prefixes and IRIs

Namespace prefix	Namespace IRI	RDF vocabulary
rdf	`http://www.w3.org/1999/02/22-rdf-syntax-ns#`	The RDF built-in vocabulary [RDF12-SCHEMA]
rdfs	`http://www.w3.org/2000/01/rdf-schema#`	The RDF Schema vocabulary [RDF12-SCHEMA]
xsd	`http://www.w3.org/2001/XMLSchema#`	The RDF-compatible XSD types

In some serialization formats it is common to abbreviate IRIs that start with namespace IRIs by using a namespace prefix in order to assist readability. For example, the IRI `http://www.w3.org/1999/02/22-rdf-syntax-ns#XMLLiteral` would be abbreviated as `rdf:XMLLiteral`. Note however that these abbreviations are *not* valid IRIs, and must not be used in contexts where IRIs are expected. Namespace IRIs and namespace prefixes are *not* a formal part of the RDF data model. They are merely a syntactic convenience for abbreviating IRIs.

The term "***namespace***" on its own does not have a well-defined meaning in the context of RDF, but is sometimes informally used to mean "namespace IRI" or "RDF vocabulary".

§ 1.5 Triple Terms and Reification to be rewritten

A triple term is an RDF term with the components of an RDF triple, which can be used as the object of another triple. It denotes a proposition, which is an abstract resource that may be true in the universe.

A triple term is NEVER ~~not necessarily~~ asserted, allowing statements to be made about other statements that may not be asserted within an RDF graph. This allows statements to be made about relationships that may be contradictory. For a triple term to be asserted, it must also appear in a graph as an asserted triple.

A triple term can be used as the object of a triple with the predicate `rdf:reifies`; such a triple is then a ***reifying triple***. The subject of that triple is called a ***reifier***. Statements can be made about such a reified proposition, using the reifier as a subject.

Concrete syntaxes, such as Turtle [RDF12-TURTLE], may have shortcuts for specifying reifying triples, capturing a triple term with its reifier, or for simultaneously asserting and reifying triples.

> NOTE
>
> Since a proposition is an abstract, logical atom, assertions are made about some reifier thereof, which can be the subject or object of different triples. This enables the description of particular circumstances, such as events, or situations, which can have dates, sources, or other contextual properties.

The following diagram represents a statement and a reification of an unasserted triple term.

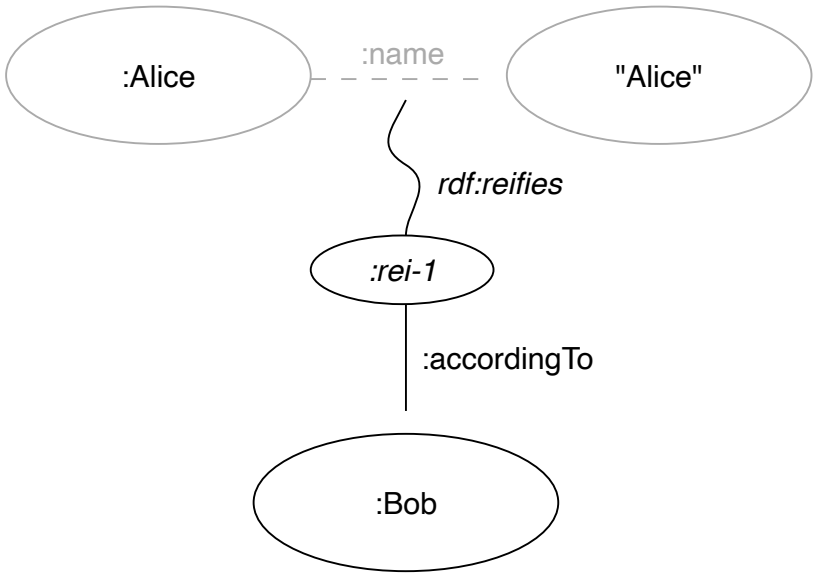

Figure 2 An RDF graph containing a triple that references an unasserted triple term (with grey dashed arc) via a reifier.

A variation on the graph shown in Figure 2 can be described where the triple term is also asserted.

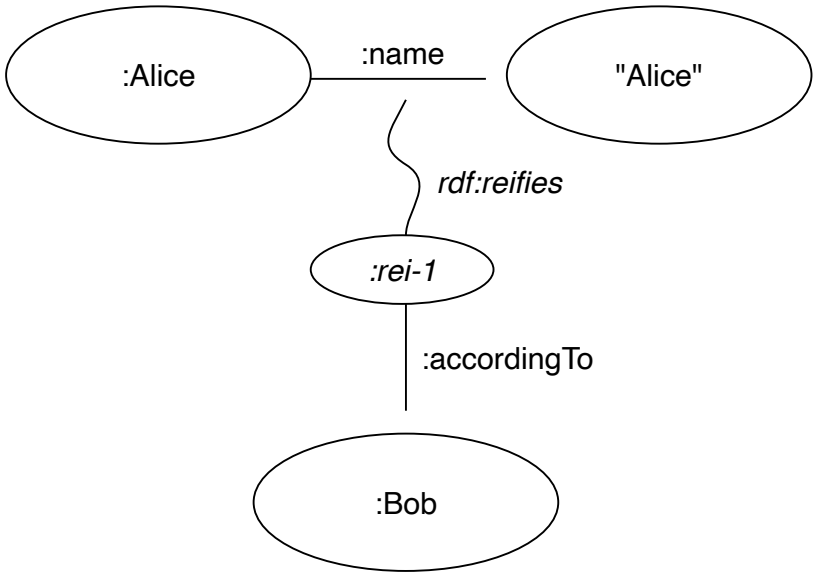

Figure 3 An RDF graph containing a triple that references an triple term, which is also asserted, via a reifier.

Note that a triple term may also have another triple term as an object.

§ 1.6 RDF and Change over Time

The RDF data model is *atemporal*: RDF graphs are static snapshots of information.

However, RDF graphs can express information about events and about temporal aspects of other entities, given appropriate vocabulary terms.

Since RDF graphs are defined as mathematical sets, adding or removing triples from an RDF graph yields a different RDF graph.

We informally use the term ***RDF source*** to refer to a persistent yet mutable source or container of RDF graphs. An RDF source is a resource that may be said to have a state that can change over time. A snapshot of the state can be expressed as an RDF graph. For example, any web document that has an RDF-bearing representation may be considered an RDF source. Like all resources, RDF sources may be named with IRIs and therefore described in other RDF graphs.

Intuitively speaking, changes in the universe of discourse can be reflected in the following ways:

- An IRI, once minted, should never change its intended referent. (See URI persistence [WEBARCH].)
- Literals, by design, are constants and never change their value.
- A relationship that holds between two resources at one time may not hold at another time.
- RDF sources may change their state over time. That is, they may provide different RDF graphs at different times.
- Some RDF sources may, however, be immutable snapshots of another RDF source, archiving its state at some point in time.

§ 1.7 Working with Multiple RDF Graphs

As RDF graphs are sets of triples, they can be combined easily, supporting the use of data from multiple sources. Nevertheless, it is sometimes desirable to work with multiple RDF graphs while keeping their contents separate. RDF datasets support this requirement.

An RDF dataset is a collection of RDF graphs. All but one of these graphs have an associated IRI or blank node. They are called named graphs, and the IRI or blank node is called the graph name. The remaining graph does not have an associated IRI, and is called the default graph of the RDF dataset.

There are many possible uses for RDF datasets. One such use is to hold snapshots of multiple RDF sources.

§ 1.8 Equivalence, Entailment and Inconsistency

An RDF triple encodes a statement—a simple ***logical expression***, or claim about the world. An RDF graph is the conjunction (logical *AND*) of its triples. The precise details of this meaning of RDF triples and graphs are the subject of *RDF 1.2 Semantics* [RDF12-SEMANTICS], which yields the following relationships between RDF graphs:

Entailment
: An RDF graph *A* entails another RDF graph *B* if every possible arrangement of the world that makes *A* true also makes *B* true. When *A* entails *B*, if the truth of *A* is presumed or demonstrated then the truth of *B* is established.

Equivalence
: Two RDF graphs *A* and *B* are equivalent if they make the same claim about the world. *A* is equivalent to *B* if and only if *A* entails *B* and *B* entails *A*.

Inconsistency
: An RDF graph is inconsistent if it contains an internal contradiction. There is no possible arrangement of the world that would make the expression true.

An ***entailment regime*** [RDF12-SEMANTICS] is a specification that defines precise conditions that make these relationships hold. RDF itself recognizes only some basic cases of entailment, equivalence and inconsistency. Other specifications, such as *RDF 1.2 Schema* [RDF12-SCHEMA] and OWL 2 [OWL2-OVERVIEW], add more powerful entailment regimes, as do some domain-specific vocabularies.

This specification does not constrain how implementations use the logical relationships defined by entailment regimes. Implementations may or may not detect inconsistencies, and may make all, some or no entailed information available to users.

§ 1.9 RDF Documents and Syntaxes

An ***RDF document*** is a document that encodes an RDF graph or RDF dataset in a ***concrete RDF syntax***, such as Turtle [RDF12-TURTLE], RDFa [RDFA-CORE], JSON-LD [JSON-LD11], or TriG [RDF12-TRIG]. RDF documents enable the exchange of RDF graphs and RDF datasets between systems.

A concrete RDF syntax may offer many different ways to encode the same RDF graph or RDF dataset, for example through the use of namespace prefixes, IRI references, blank node identifiers, and different ordering of triples. While these aspects can have great effect on the convenience of working with the RDF document, they are not significant for its meaning.

§ 2. Conformance

As well as sections marked as non-normative, all authoring guidelines, diagrams, examples, and notes in this specification are non-normative. Everything else in this specification is normative.

The key words *MAY*, *MUST*, *MUST NOT*, *RECOMMENDED*, *SHOULD*, and *SHOULD NOT* in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This specification, *RDF 1.2 Concepts and Abstract Syntax*, defines a data model and related terminology for use in other specifications, such as concrete RDF syntaxes, API specifications, and query languages. Implementations cannot directly conform to *RDF 1.2 Concepts and Abstract Syntax*, but can conform to such other specifications that normatively reference terms defined here.

This specification establishes two conformance levels:

- ***Full conformance*** supports graphs and datasets with triples that contain triple terms. Concrete syntaxes in which such graphs and datasets can be expressed include [RDF12-N-TRIPLES], [RDF12-N-QUADS], [RDF12-TURTLE], and [RDF12-TRIG].
- ***Classic conformance*** only supports graphs or datasets with triples that do not contain triple terms.

> EDITOR'S NOTE
>
> The conformance levels described above are tentative, and still the subject of group discussion. An alternative to conformance levels, "profiles", may be adopted instead, abandoned, or described in another specification.

2.1 Strings in RDF

RDF uses Unicode [Unicode] as the fundamental representation for string values. Within this, and related specifications, the term ***string***, or RDF string, is used to describe an ordered sequence of zero or more Unicode code points which are Unicode scalar values. Unicode scalar values do not include the surrogate code points. Note that most concrete RDF syntaxes require the use of the UTF-8 character encoding [RFC3629], and use the `\u0000` or `\U00000000` forms to express certain non-character values.

A string is identical to another string if it consists of the same sequence of code points. An implementation *MAY* determine string equality by comparing the code units of two strings that use the same Unicode character encoding (UTF-8 or UTF-16) without decoding the string into a Unicode code point sequence.

3. RDF Graphs

An ***RDF graph*** is a set of RDF triples.

An RDF triple is said to be ***asserted*** in an RDF graph if it is an element of the RDF graph.

3.1 Triples

An ***RDF triple*** (usually called "triple") is a 3-tuple (s, p, o) with the following characteristics:

- s is an IRI or a blank node.
- p is an IRI.

- *o* is an IRI, a blank node, a literal, or a triple term.

IRIs, literals, blank nodes, and triple terms are collectively known as ***RDF terms***.

IRIs, literals, blank nodes, and triple terms are distinct and distinguishable. For example, a literal with the string `http://example.org/` as its lexical form is not equal to the IRI `http://example.org/`, nor to a blank node with the blank node identifier `http://example.org/`.

The set of ***nodes*** of an RDF graph is the set of subjects and objects of the triples in the graph. It is possible for a predicate IRI to also occur as a node in the same graph.

§ 3.2 IRIs

An ***IRI*** (Internationalized Resource Identifier) within an RDF graph is a string that conforms to the syntax defined in RFC 3987 [RFC3987].

> NOTE
>
> For convenience, a complete [ABNF] grammar from [RFC3987] is provided in E. IRI Grammar.

IRIs in the RDF abstract syntax *MUST* be resolved per [RFC3986], and *MAY* contain a fragment identifier.

IRI equality: Two IRIs are the same if and only if they consist of the same sequence of Unicode code points, as in Simple String Comparison in section 5.3.1 of [RFC3987]. (This is done in the abstract syntax, so the IRIs are resolved IRIs with no escaping or encoding.) Further normalization *MUST NOT* be performed before this comparison.

> NOTE
>
> **URIs and IRIs:** IRIs are a generalization of ***URIs*** [RFC3986] that permits a wider range of Unicode characters [UNICODE]. Every URI and URL is an IRI, but not every IRI is an URI. In RDF, IRIs are used as ***IRI references***, as defined in [RFC3987] section 1.3. An IRI reference is common usage of an Internationalized Resource Identifier. An IRI reference refers to either a resolved IRI or relative IRI reference, as described by the *IRI-reference* production in E. IRI Grammar. The abstract syntax uses only fully resolved IRIs. When IRIs are used in operations that are only defined for URIs, they

must first be converted according to the mapping defined in section 3.1 of [RFC3987]. A notable example is retrieval over the HTTP protocol. The mapping involves UTF-8 encoding of non-ASCII characters, %-encoding of octets not allowed in URIs, and Punycode-encoding of domain names.

URLs: The *URL Standard* is largely compatible with [RFC3987] IRIs, but is based on a processing model important for implementation within web browsers and are not described using an [ABNF] grammar.

Relative IRI references: Some concrete RDF syntaxes permit ***relative IRI references*** as a convenient shorthand that allows authoring of documents independently from their final publishing location. Relative IRI references must be resolved against a ***base IRI***. Therefore, the RDF graph serialized in such syntaxes is well-defined only if a base IRI can be established [RFC3986].

IRI normalization: Interoperability problems can be avoided by minting only IRIs that are normalized according to Section 5 of [RFC3987]. Non-normalized forms that are best avoided include the following:

- Uppercase characters in scheme names and domain names
- Percent-encoding of characters where it is not required by IRI syntax
- Explicit inclusion of the HTTP default port (`http://example.com:80/`); `http://example.com/` is preferable
- A completely empty path in an HTTP IRI (`http://example.com`); `http://example.com/` is preferable
- "`/./`" or "`/../`" in the path component of an IRI
- Lowercase hexadecimal letters within percent-encoding triplets (i.e., "`%3F`" is preferred over "`%3f`")
- Punycode-encoding of Internationalized Domain Names in IRIs [RFC3492]
- IRIs that are not in Unicode Normalization Form C [I18N-Glossary]

§ 3.3 Literals

Literals are used for values such as strings, numbers, and dates.

A ***literal*** in an RDF graph consists of two, three, or four elements, as follow:

1. a ***lexical form*** consisting of a sequence of Unicode code points [UNICODE] which are Unicode scalar values, and therefore do not contain Unicode surrogate code points
2. a ***datatype IRI***, being an IRI identifying a datatype that determines how the lexical form maps to a literal value
3. if and only if the datatype IRI is `http://www.w3.org/1999/02/22-rdf-syntax-ns#langString`, a non-empty ***language tag*** as defined by [BCP47]. The language tag *MUST* be well-formed according to section 2.2.9 of [BCP47], and *MUST* be treated consistently, that is, in a case insensitive manner. Two language tags are the same if they only differ by case.
4. if and only if the datatype IRI is `http://www.w3.org/1999/02/22-rdf-syntax-ns#dirLangString`, a non-empty language tag that *MUST* be well-formed according to section 2.2.9 of [BCP47], and *MUST* be treated consistently, that is, in a case insensitive manner, and a ***base direction*** that *MUST* be either `ltr` or `rtl`.

A literal is a ***language-tagged string*** if the third element is present and the fourth element is not present. Lexical representations of language tags *MAY* be case normalized, (for example, by canonicalizing as defined by BCP 47 section 4.5).

A literal is a ***directional language-tagged string*** if both the third element and fourth elements are present. The third element, the language tag, is treated identically as in a language-tagged string, and the fourth element, base direction, *MUST* be either `ltr` or `rtl`, which *MUST* be in lower case.

The meanings of the base direction values are:

- `ltr`: indicates that the initial text direction is set to left-to-right.
- `rtl`: indicates that the initial text direction is set to right-to-left.

Please note that concrete syntaxes *MAY* support ***simple literals*** consisting of only a lexical form without any datatype IRI, language tag, or base direction. Simple literals are syntactic sugar for abstract syntax literals with the datatype IRI `http://www.w3.org/2001/XMLSchema#string` (which is commonly abbreviated as `xsd:string`). Similarly, most concrete syntaxes represent language-tagged strings and directional language-tagged strings without the datatype IRI because it always equals either `http://www.w3.org/1999/02/22-rdf-syntax-ns#langString` (`rdf:langString`) or `http://www.w3.org/1999/02/22-rdf-syntax-ns#dirLangString` (`rdf:dirLangString`), respectively.

The ***literal value*** associated with a literal is:

- If the literal is a language-tagged string, then the literal value is a pair consisting of its lexical form and its language tag, in that order.
- If the literal is a directional language-tagged string, then the literal value is a tuple of its lexical form, its language tag, and its base direction, likewise in that order.
- If the literal's datatype is handled by an RDF implmentation,
    - if the literal's lexical form is in the lexical space of the datatype, then the literal value is the result of applying the lexical-to-value mapping of the datatype to the lexical form.
    - otherwise, the literal is ill-typed and no literal value can be associated with the literal. Such a case produces a semantic inconsistency but is not *syntactically* ill-formed. Implementations *SHOULD* accept ill-typed literals and produce RDF graphs from them. Implementations *MAY* produce warnings when encountering ill-typed literals.
- If the literal's datatype IRI is *not* handled by an RDF implementation, then the literal value is not defined by this specification.

Literal term equality: Two literals are term-equal (the same RDF literal) if and only if:

- the two lexical forms compare equal
- the two datatype IRIs compare equal
- the two language tags (if any) compare equal
- the two base directions (if any) compare equal

Comparison is performed using case sensitive matching (see description of string comparison in 2.1 Strings in RDF) except for language tags, where the comparison is performed using ASCII case-insensitive matching. Thus, two literals can have the same value without being the same RDF term. For example:

```
"1"^^xs:integer
"01"^^xs:integer
```

denote the same value, but are not the same literal RDF terms and are not term-equal because their lexical forms differ.

§ 3.4 Blank Nodes

Blank nodes are disjoint from IRIs and literals. Otherwise, the set of possible blank nodes is arbitrary. RDF makes no reference to any internal structure of blank nodes.

> NOTE
>
> ***Blank node identifiers*** are local identifiers that are used in some concrete RDF syntaxes or RDF store implementations. They are always locally scoped to the file or RDF store, and are *not* persistent or portable identifiers for blank nodes. Blank node identifiers are *not* part of the RDF abstract syntax, but are entirely dependent on the concrete syntax or implementation. The syntactic restrictions on blank node identifiers, if any, therefore also depend on the concrete RDF syntax or implementation. Implementations that handle blank node identifiers in concrete syntaxes need to be careful not to create the same blank node from multiple occurrences of the same blank node identifier except in situations where this is supported by the syntax.
>
> The term "blank node label" is sometimes used informally as an alternative to the term blank node identifier. This alternative was also used in earlier versions of some RDF-related specifications such as [SPARQL11-QUERY]. In the interest of consistency, the use of this alternative term is discouraged now.

§ 3.5 Triple Terms

A ***triple term*** is a 3-tuple that is defined recursively as follows:

- If s is an IRI or a blank node, p is an IRI, and o is an IRI, a blank node, a literal, or a triple term, then (s, p, o) is a triple term.

Given a triple (s, p, o), s is called the ***subject*** of the triple, p is called the ***predicate*** of the triple, and o is called the ***object*** of the triple. Similarly, given a triple term (s, p, o), s is called the *subject* of the triple term, p is called the *predicate* of the triple term, and o is called the *object* of the triple term.

A

~~The only~~ difference between a triple and a triple term is that a triple can be a member of a graph, and a triple term can be used as the object of another triple or triple term.

> NOTE
>
> While, syntactically, the notion of an RDF triple and the notion of a triple term are the same, they represent different concepts. RDF triples are the members of RDF graphs, whereas triple terms can be used as components of RDF triples.

> NOTE
>
> The definition of triple term is recursive. That is, a triple term can itself have an object component which is another triple term. However, by this definition, cycles of triple terms cannot be created.

> NOTE
>
> ~~Every triple with a triple term as its object *SHOULD* use `http://www.w3.org/1999/02/22-rdf-syntax-ns#reifies` (`rdf:reifies`) as its predicate. Every triple whose object is not a triple term *SHOULD NOT* use `http://www.w3.org/1999/02/22-rdf-syntax-ns#reifies` (`rdf:reifies`) as its predicate.~~

NEW SECTION HERE ON “REIFICATION” WITH RDF:REIFIES

§ 3.6 Initial Text Direction

This section is non-normative.

The base direction of a directional language-tagged string provides a means of establishing the initial direction of text, including text which is a mixture of right-to-left and left-to-right scripts. The Unicode Bidirectional Algorithm [I18N-Glossary] provides support for automatically rendering a sequence of characters in logical order, so that they are visually ordered as expected, but this is not sufficient to correctly render bidirectional text.

For example, some text with a language tag "`he`" might be displayed in a left-to-right context (such as an English web page) as פעילות הבינאום, W3C, which is incorrect. When provided to a user agent including base direction information (such as using HTML's `dir` attribute) it can then be correctly presented as:

In the absence of an explicit initial text direction, the Unicode Bidirectional Algorithm detects the text direction from the content. This depends on the first strongly directional character in the text or on the context. To avoid spillover effects, users need to employ bidi isolation whenever text is inserted into a larger document. For example, "<bdi lang="he">ספרים בינלאומיים!</bdi>" displays the Hebrew characters in a right-to-left fashion — i.e., as "ספרים בינלאומיים!" — while they would be stored in memory as "מירפס מיימואלניב!"

§ 3.7 Replacing Blank Nodes with IRIs

```
:a :b _:x. ⊨ :a :b _:y.
:a :b  :x. ⊭ :a :b  :y.
```

Blank nodes do not have identifiers in the RDF abstract syntax. The blank node identifiers introduced by some concrete syntaxes have only local scope and are purely an artifact of the serialization.

In situations where stronger identification is needed, systems *MAY* systematically replace some or all of the blank nodes in an RDF graph with IRIs. Systems wishing to do this *SHOULD* mint a new, globally unique IRI (a ***Skolem IRI***) for each blank node so replaced.

WRONG!

This transformation does not appreciably change the meaning of an RDF graph, provided that the Skolem IRIs do not occur anywhere else. It does however permit the possibility of other graphs subsequently using the Skolem IRIs, which is not possible for blank nodes.

Systems may wish to mint Skolem IRIs in such a way that they can recognize the IRIs as having been introduced solely to replace blank nodes. This allows a system to map IRIs back to blank nodes if needed.

Systems that want Skolem IRIs to be recognizable outside of the system boundaries *SHOULD* use a well-known IRI [RFC8615] with the registered name `genid`. This is an IRI that uses the HTTP or HTTPS scheme, or another scheme that has been specified to use well-known IRIs; and whose path component starts with `/.well-known/genid/`.

For example, the authority responsible for the domain `example.com` could mint the following recognizable Skolem IRI:

```
http://example.com/.well-known/genid/d26a2d0e98334696f4ad70a677abc1
```

> NOTE
>
> RFC 8615 [RFC8615] only specifies well-known URIs, not IRIs. For the purpose of this document, a well-known IRI is any IRI that results in a well-known URI after IRI-to-URI mapping [RFC3987].

3.8 Graph Comparison

Two RDF graphs *G* and *G'* are ***isomorphic*** (that is, they have an identical form) if there is a bijection *M* between the sets of nodes of the two graphs, such that all of the following properties hold:

- *M* maps blank nodes to blank nodes.
- *M(lit)=lit* for all RDF literals *lit* which are nodes of *G*.
- *M(iri)=iri* for all IRIs *iri* which are nodes of *G*.
- The triple *(s, p, o)* is in *G* if and only if the triple *(M(s), p, M(o))* is in *G'*

See also: IRI equality, literal term equality.

With this definition, *M* shows how each blank node in *G* can be replaced with a new blank node to give *G'*. Graph isomorphism is needed to support the RDF Test Cases [RDF11-TESTCASES] specification.

4. RDF Datasets

An ***RDF dataset*** is a collection of RDF graphs, and comprises:

- Exactly one ***default graph***, being an RDF graph. The default graph does not have a name and *MAY* be empty.
- Zero or more ***named graphs***. Each named graph is a pair consisting of an IRI or a blank node (the ***graph name***), and an RDF graph. Graph names are unique within an RDF dataset.

Blank nodes can be shared between graphs in an RDF dataset.

NOTE

Despite the use of the word "name" in "named graph", the graph name is not required to denote the graph. It is merely syntactically paired with the graph. RDF does not place any formal restrictions on what resource the graph name may denote, nor on the relationship between that resource and the graph. A discussion of different RDF dataset semantics can be found in [RDF11-DATASETS].

Some RDF dataset implementations do not track empty named graphs. Applications can avoid interoperability issues by not ascribing importance to the presence or absence of empty named graphs.

SPARQL 1.2 [SPARQL12-CONCEPTS] also defines the concept of an RDF Dataset. The definition of an RDF Dataset in SPARQL 1.1 and this specification differ slightly in that this specification allows RDF Graphs to be identified using either an IRI or a blank node. SPARQL 1.1 Query Language only allows RDF Graphs to be identified using an IRI. Existing SPARQL implementations might not allow blank nodes to be used to identify RDF Graphs for some time, so their use can cause interoperability problems. Skolemizing blank nodes used as graph names can be used to overcome these interoperability problems.

§ 4.1 RDF Dataset Comparison

Two RDF datasets (the RDF dataset *D1* with default graph *DG1* and any named graph *NG1* and the RDF dataset *D2* with default graph *DG2* and any named graph *NG2*) are ***dataset-isomorphic*** if and only if there is a bijection *M* between the nodes, triples and graphs in *D1* and those in *D2* such that all of the following properties hold:

- *M* maps blank nodes to blank nodes;
- *M* is the identity map on literals and URIs;
- For every triple <s p o>, *M*(<s, p, o>)= <*M(s)*, *M(p)*, *M(o)*>;
- For every graph *G*={t1, ..., tn}, *M(G)*={*M(t1)*, ..., *M(tn)*};
- *DG2* = *M(DG1)*; and
- <n, G> is in *NG1* if and only if <*M(n)*, *M(G)*> is in *NG2*.

§ 4.2 Content Negotiation of RDF Datasets

This section is non-normative.

Web resources may have multiple representations that are made available via content negotiation [WEBARCH]. A representation may be returned in an RDF serialization format that supports the expression of both RDF datasets and RDF graphs. If an RDF dataset is returned and the consumer is expecting an RDF graph, the consumer is expected to use the RDF dataset's default graph.

§ 4.3 Dataset as a Set of Quads

This section is non-normative.

A ***quad*** is a triple associated with an optional graph name and is used when referring to triples within an RDF dataset.

A quad can be represented as a tuple composed of subject, predicate, object, and an optional graph name.

An RDF dataset can be considered to be a set of quads where quads with no graph name supply the triples of the default graph, and quads with the same graph name supply the triples of the named graph with that name.

> NOTE
>
> Although a quad without a graph name consists of the same three components as a triple, it is a distinct concept, as it specifically captures the notion of a triple within the default graph of an RDF dataset.

§ 5. Datatypes

Datatypes are used with RDF literals to represent values such as strings, numbers and dates. The datatype abstraction used in RDF is compatible with XML Schema [XMLSCHEMA11-

2]. Any datatype definition that conforms to this abstraction *MAY* be used in RDF, even if not defined in terms of XML Schema. RDF re-uses many of the XML Schema built-in datatypes, and defines three additional datatypes, `rdf:JSON`, `rdf:HTML`, and `rdf:XMLLiteral`.

A ***datatype*** consists of a lexical space, a value space and a lexical-to-value mapping, and is identified by one or more IRIs.

The ***lexical space*** of a datatype is a set of strings.

The ***lexical-to-value mapping*** of a datatype is a set of pairs whose first element belongs to the lexical space, and the second element belongs to the ***value space*** of the datatype. Each member of the lexical space is paired with exactly one value, and is a *lexical representation* of that value. The mapping can be seen as a function from the lexical space to the value space.

> NOTE
>
> Language-tagged strings have the datatype IRI `http://www.w3.org/1999/02/22-rdf-syntax-ns#langString` (commonly abbreviated as `rdf:langString`). No datatype is formally defined for this IRI because the definition of datatypes does not accommodate language tags in the lexical space. The value space associated with this datatype IRI is the set of all pairs that consist of a string and a language tag. Similarly, directional language-tagged strings `http://www.w3.org/1999/02/22-rdf-syntax-ns#dirLangString` (commonly abbreviated as `rdf:dirLangString`) also have a base direction in the value space. The value space associated with this datatype IRI is the set of all 3-tuples of a string, a language tag and a base direction.

For example, the XML Schema datatype `xsd:boolean`, where each member of the value space has two lexical representations, is defined as follows:

Lexical space:
{“`true`”, “`false`”, “`1`”, “`0`”}

Value space:
{***true***, ***false***}

Lexical-to-value mapping
{ <“`true`”, ***true***>, <“`false`”, ***false***>, <“`1`”, ***true***>, <“`0`”, ***false***>, }

The literals that can be defined using this datatype are:

This table lists the literals of type xsd:boolean.

Literal	Value
<“`true`”, `xsd:boolean`>	***true***
<“`false`”, `xsd:boolean`>	***false***
<“`1`”, `xsd:boolean`>	***true***
<“`0`”, `xsd:boolean`>	***false***

§ 5.1 The XML Schema Built-in Datatypes

IRIs of the form `http://www.w3.org/2001/XMLSchema#xxx`, where *`xxx`* is the name of a datatype, denote the built-in datatypes defined in *W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes* [XMLSCHEMA11-2]. The XML Schema built-in types listed in the following table are the ***RDF-compatible XSD types***. Their use is *RECOMMENDED*.

Readers might note that the only safe datatypes for transferring binary information are `xsd:hexBinary` and `xsd:base64Binary`.

A list of the RDF-compatible XSD types, with short descriptions

	Datatype	Value space (informative)
Core types	`xsd:string`	Character strings
	`xsd:boolean`	true, false
	`xsd:decimal`	Arbitrary-precision decimal numbers
	`xsd:integer`	Arbitrary-size integer numbers
IEEE floating-point numbers	`xsd:double`	64-bit floating point numbers incl. ±Inf, ±0, NaN
	`xsd:float`	32-bit floating point numbers incl. ±Inf, ±0, NaN
Time and date	`xsd:date`	Dates (yyyy-mm-dd) with or without timezone
	`xsd:time`	Times (hh:mm:ss.sss…) with or without timezone
	`xsd:dateTime`	Date and time with or without

		timezone
	xsd:dateTimeStamp	Date and time with required timezone
Recurring and partial dates	xsd:gYear	Gregorian calendar year
	xsd:gMonth	Gregorian calendar month
	xsd:gDay	Gregorian calendar day of the month
	xsd:gYearMonth	Gregorian calendar year and month
	xsd:gMonthDay	Gregorian calendar month and day
	xsd:duration	Duration of time
	xsd:yearMonthDuration	Duration of time (months and years only)
	xsd:dayTimeDuration	Duration of time (days, hours, minutes, seconds only)
Limited-range integer numbers	xsd:byte	-128…+127 (8 bit)
	xsd:short	-32768…+32767 (16 bit)
	xsd:int	-2147483648…+2147483647 (32 bit)
	xsd:long	-9223372036854775808… +9223372036854775807 (64 bit)
	xsd:unsignedByte	0…255 (8 bit)
	xsd:unsignedShort	0…65535 (16 bit)
	xsd:unsignedInt	0…4294967295 (32 bit)
	xsd:unsignedLong	0…18446744073709551615 (64 bit)
	xsd:positiveInteger	Integer numbers >0
	xsd:nonNegativeInteger	Integer numbers ≥0
	xsd:negativeInteger	Integer numbers <0
	xsd:nonPositiveInteger	Integer numbers ≤0
Encoded	xsd:hexBinary	Hex-encoded binary data

binary data	`xsd:base64Binary`	Base64-encoded binary data
Miscellaneous XSD types	`xsd:anyURI`	Resolved or relative URI and IRI references
	`xsd:language`	Language tags per [BCP47]
	`xsd:normalizedString`	Whitespace-normalized strings
	`xsd:token`	Tokenized strings
	`xsd:NMTOKEN`	XML NMTOKENs
	`xsd:Name`	XML Names
	`xsd:NCName`	XML NCNames

The lexical-to-value mapping for `xsd:float` and `xsd:double` *MUST* use a method consistent with doubleLexicalMap, which *MUST* strictly conform to the rounding method described in floatPtRound [XMLSCHEMA11-2].

The other built-in XML Schema datatypes are unsuitable for various reasons and *SHOULD NOT* be used:

- `xsd:QName` and `xsd:ENTITY` require an enclosing XML document context.
- `xsd:ID` and `xsd:IDREF` are for cross references within an XML document.
- `xsd:NOTATION` is not intended for direct use.
- `xsd:IDREFS`, `xsd:ENTITIES` and `xsd:NMTOKENS` are sequence-valued datatypes which do not fit the RDF datatype model.

> NOTE
>
> The value spaces of `xsd:double` and `xsd:float` do not include all decimal numbers. For every literal of either of these two datatypes, the value of the literal is a value that can be represented as an IEEE 754-2008 binary floating point representation of the corresponding precision. For instance, the literal with lexical form `"0.1"` and datatype `xsd:float` denotes the number `0.100000001490116119384765625`. Rather than `xsd:double` or `xsd:float`, the datatype `xsd:decimal` can be used to accurately capture arbitrary decimal numbers.

§ 5.2 Datatype IRIs

Datatypes are identified by IRIs.

If any IRI of the form `http://www.w3.org/2001/XMLSchema#xxx` is handled by an RDF implementation, it *MUST* refer to the RDF-compatible XSD type named `xsd:xxx` for every XSD type listed in section 5.1.

The datatypes identified by the three IRIs below are defined in Appendix A. Additional Datatypes:

- The IRI `http://www.w3.org/1999/02/22-rdf-syntax-ns#XMLLiteral` refers to the datatype `rdf:XMLLiteral`.
- The IRI `http://www.w3.org/1999/02/22-rdf-syntax-ns#HTML` refers to the datatype `rdf:HTML`.
- The IRI `http://www.w3.org/1999/02/22-rdf-syntax-ns#JSON` refers to the datatype `rdf:JSON`.

RDF implementations are not required to handle all datatypes. Any literal typed with a datatype not handled by an RDF implementation is treated just like an unknown IRI, i.e., as referring to an unknown thing. Applications *MAY* give a warning message if they are unable to determine the referent of an IRI used in a typed literal. RDF implementations *SHOULD* not reject a literal with an unknown datatype as either a syntactic or semantic error.

Other specifications *MAY* impose additional constraints on datatype IRIs, for example, require support for certain datatypes.

> NOTE
>
> Semantic extensions of RDF might choose to recognize other datatype IRIs and require each of them to refer to a fixed datatype. See *RDF 1.2 Semantics* [RDF12-SEMANTICS] for more information on semantic extensions.

NOTE

The Web Ontology Language [OWL2-OVERVIEW] offers facilities for formally defining custom datatypes that can be used with RDF. Furthermore, a practice for identifying user-defined simple XML Schema datatypes is suggested in [SWBP-XSCH-DATATYPES]. RDF implementations are not required to support either of these facilities.

NOTE

In RDF 1.1, *Recognized datatype IRIs* were defined in RDF Concepts, overlapping with RDF Semantics, "recognizing" datatype IRIs for semantic extensions.

§ 6. Fragment Identifiers

This section is non-normative.

RDF uses IRIs, which may include ***fragment identifiers***, as resource identifiers. The semantics of fragment identifiers is defined in RFC 3986 [RFC3986]: They identify a secondary resource that is usually a part of, a view of, defined in, or described in the primary resource, and the precise semantics depend on the set of representations that might result from a retrieval action on the primary resource.

This section discusses the handling of fragment identifiers in representations that encode RDF graphs.

In RDF-bearing representations of a primary resource, e.g., `<https://example.com/foo>`, the secondary resource identified by a fragment identifier, e.g., `bar`, is the resource denoted by the full IRI in the RDF graph, which would be `<https://example.com/foo#bar>` in this case. Since IRIs in RDF graphs can denote anything, this can be something external to the representation, or even external to the web.

In this way, the RDF-bearing representation acts as an intermediary between the web-accessible primary resource, and some set of possibly non-web or abstract entities that the RDF graph may describe.

In cases where other specifications constrain the semantics of fragment identifiers in RDF-bearing representations, the encoded RDF graph should use fragment identifiers in a way

that is consistent with these constraints. For example, in an HTML+RDFa document [HTML-RDFA], a fragment identifier such as `chapter1` may identify a document section via the semantics of HTML's `@name` or `@id` attributes. Such an IRI, e.g., `<#chapter1>`, should then be taken to denote that same section in any RDFa-encoded triples within the same document. Similarly, fragment identifiers should be used consistently in resources with multiple representations that are made available via content negotiation [WEBARCH]. For example, if the fragment identifier `chapter1` identifies a document section in an HTML representation of the primary resource, then the IRI `<#chapter1>` should be taken to denote that same section in all RDF-bearing representations of the same primary resource.

§ 7. Generalizations of RDF Triples, Graphs, and Datasets

This section is non-normative.

It is sometimes convenient to loosen the requirements on RDF triples. For example, the completeness of the RDFS entailment rules is easier to show with a notion of symmetric RDF triples. `Why people should care about this??`

`what about symmetric triple terms?`

A ***symmetric RDF triple*** allows the subject to be any RDF term that is allowed in the object position, one of an IRI, a blank node, a literal or a triple term. A ***symmetric RDF graph*** is a set of symmetric RDF triples. A ***symmetric RDF dataset*** comprises a distinguished symmetric RDF graph, and zero or more pairs that each associate an IRI or a blank node with a symmetric RDF graph.

Symmetric RDF triples, graphs, and datasets differ from standard normative RDF triples, graphs, and datasets only by allowing IRIs, blank nodes, literals, or triple terms in the subject and object positions.

`what about generalised triple terms?`

A ***generalized RDF triple*** is a triple having a subject, a predicate, and an object, where each can be an IRI, a blank node, a triple term, or a literal. A ***generalized RDF graph*** is a set of generalized RDF triples. A ***generalized RDF dataset*** comprises a distinguished generalized RDF graph, and zero or more pairs each associating an IRI, a blank node, a triple term, or a literal to a generalized RDF graph.

Generalized RDF triples, graphs, and datasets differ from standard normative RDF triples, graphs, and datasets only by allowing IRIs, blank nodes, triple terms, and literals to appear in any position, i.e., as subject, predicate, object, or graph name.

Why? This is non-normative, just like so many parts of the spec. Why say this only here?

NOTE

Any user of symmetric or generalized RDF triples, graphs, or datasets needs to be aware that these notions are non-standard extensions of RDF, and their use may cause interoperability problems. There is no requirement for any RDF tool to accept, process, or produce anything beyond standard normative RDF triples, graphs, and datasets.

A. Additional Datatypes

This section defines additional datatypes that RDF implementations *MAY* support.

A.1 The `rdf:HTML` Datatype

RDF provides for HTML content as a possible literal value. This allows markup in literal values. Such content is indicated in an RDF graph using a literal whose datatype is set to ***`rdf:HTML`***.

The `rdf:HTML` datatype is defined as follows:

The IRI denoting this datatype
is `http://www.w3.org/1999/02/22-rdf-syntax-ns#HTML`.

The value space
is the set of DOM `DocumentFragment` nodes [DOM]. Two `DocumentFragment` nodes *node* and *otherNode* are considered equal if and only if the DOM method `node.isEqualNode(otherNode)` [DOM] returns `true`.

The lexical-to-value mapping
Each member of the lexical space is associated with the result of applying the following algorithm:

- Let `domnodes` be the list of DOM nodes [DOM] that result from applying the HTML fragment parsing algorithm [HTML5] to the input string, without a context element.
- Let `domfrag` be a DOM `DocumentFragment` [DOM] whose `childNodes` attribute is equal to `domnodes`
- Return `domfrag.normalize()`.

NOTE

Any language annotation (`lang="…"`), text directionality annotation (`dir="…"`), or XML namespaces (`xmlns`) desired in the HTML content must be included explicitly in the HTML literal. Relative URLs in attributes such as `href` do not have a well-defined base URL and are best avoided. RDF applications may use additional equivalence relations, such as that which relates an `xsd:string` with an `rdf:HTML` literal corresponding to a single text node of the same string.

§ A.2 The `rdf:XMLLiteral` Datatype

RDF provides for XML content as a possible literal value. Such content is indicated in an RDF graph using a literal whose datatype is set to ***`rdf:XMLLiteral`***.

The `rdf:XMLLiteral` datatype is defined as follows:

The IRI denoting this datatype
is `http://www.w3.org/1999/02/22-rdf-syntax-ns#XMLLiteral`.

The lexical space
is the set of all strings which are well-balanced, self-contained XML content [XML11]; and for which embedding between an arbitrary XML start tag and an end tag yields a document conforming to *Namespaces in XML 1.0 (Third Edition)* [XML-NAMES].

The value space
is the set of DOM `DocumentFragment` nodes [DOM]. Two `DocumentFragment` nodes *node* and *otherNode* are considered equal if and only if the DOM method *`node`*`.isEqualNode(`*`otherNode`*`)` returns `true`.

The lexical-to-value mapping
Each member of the lexical space is associated with the result of applying the following algorithm:

- Let `domfrag` be a DOM `DocumentFragment` node [DOM] corresponding to the input string.
- Return `domfrag.normalize()`.

NOTE

Any XML namespace declarations (`xmlns`), language annotation (`xml:lang`) or base URI declarations (`xml:base`) desired in the XML content must be included explicitly in the XML literal. Note that some concrete RDF syntaxes may define mechanisms for inheriting them from the context (e.g., `@parseType="literal"` in RDF/XML [RDF12-XML].

§ A.3 The `rdf:JSON` Datatype

RDF provides for JSON content as a possible literal value. This includes allowing markup in literal values. Such content is indicated in an RDF graph as a literal whose datatype is set to ***`rdf:JSON`***.

The `rdf:JSON` datatype is defined as follows:

The IRI denoting this datatype
is `http://www.w3.org/1999/02/22-rdf-syntax-ns#JSON`.

The lexical space
is the set of all RDF strings that conform to the JSON Grammar as described in Section 2 JSON Grammar of [RFC8259], which also conform to the requirements of *The I-JSON Message Format* [RFC7493].

> NOTE
> *The JavaScript Object Notation (JSON) Data Interchange Format* [RFC8259] allows strings to include surrogate code points not allowed in RDF strings, which are also excluded in [RFC7493], thus the lexical representation of JSON literals excludes those including surrogate code points.

The value space
is the smallest set containing strings, numbers (**xsd:double**), maps (mapping strings to values in the value space where the order of map entries is not significant), lists (of values in the value space), and literal values (`true`, `false`, and `null`) from *Infra Standard* [INFRA] and *W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes* [XMLSCHEMA11-2].

> NOTE
>
> The value space of maps and lists does not include values having themselves as members, which cannot be represented in JSON.

Two values (a and b) are considered equal if any of the following are true:

- They are the same string, number (**xsd:double**), or literal value.
- They are both lists containing items which are pairwise equal – meaning that each item in a is equal the item at the corresponding index in b, and both a and b have the same size.
- They are both maps with equal entries – meaning that for each entry e_a in a there exists an entry e_b in b such that the key in e_a equals the key in e_b, the value in e_a equals the value in e_b, and both a and b have the same size.

  > NOTE
  > Two JSON Objects containing maps which are serialized with entries in a different order will be equal under this definition when transformed to the value space. For example, `{ "a": 1, "b": 2 }` and `{ "b": 2, "a": 1 }` are considered equal. As a result of the value space being defined using terminology from [INFRA], property values which can contain more than one item, such as lists and maps, are explicitly ordered. All list-like value structures in [INFRA] are ordered, whether or not that order is significant. For the purposes of this specification, unless otherwise stated, map ordering is not important and implementations are not expected to produce or consume deterministically ordered values.

The lexical-to-value mapping
maps every element of the lexical space to the result of parsing it into a string, number (**xsd:double**), map, list, or literal value (`true`, `false`, and `null`).

- A JSON Object is mapped to a map by transforming each object member into a map entry with the key taken from the member name and value taken by performing this mapping to the member value. Map entries are treated as being unordered.
- A JSON Array is mapped to a list such that this list contains as many elements as the JSON Array and, for every position i in the array, the element at the i-th position in the list is the value that results from applying this mapping to the i-th element of the array.

- A JSON Number is mapped to an **xsd:double** using a method consistent with doubleLexicalMap, which *MUST* strictly conform to the rounding method described in floatPtRound [XMLSCHEMA11-2].

  > NOTE
  > Some numbers cannot be represented as finite **xsd:double** values and may map to `+INF` or `-INF`. Such values cannot be represented as JSON Numbers, limiting the ability to serialize such values back to JSON.

- A JSON String is mapped to a string after converting any escape sequences to the associated Unicode code point.
- A JSON literal name maps JSON `true`, `false`, and `null` values to [INFRA] `true`, `false`, and `null` values, respectively.

§ B. Privacy Considerations

This section is non-normative.

RDF is used to express arbitrary application data, which may include the expression of personally identifiable information (PII) or other information which could be considered sensitive. Authors publishing such information are advised to carefully consider the needs and use of publishing such information, as well as the applicable regulations for the regions where the data is expected to be consumed and potentially revealed (e.g., GDPR, CCPA, others), particularly whether authorization measures are needed for access to the data.

§ C. Security Considerations

This section is non-normative.

The RDF Abstract Syntax is not used directly for conveying information, although concrete serialization forms are specifically intended to do so.

Applications *MAY* evaluate given data to infer more assertions or to dereference IRIs, invoking the security considerations of the scheme for that IRI. Note in particular, the privacy issues in [RFC3023] section 10 for HTTP IRIs. Data obtained from an inaccurate or

malicious data source may lead to inaccurate or misleading conclusions, as well as the dereferencing of unintended IRIs. Care must be taken to align the trust in consulted resources with the sensitivity of the intended use of the data; inferences of potential medical treatments would likely require different trust than inferences for trip planning.

RDF is used to express arbitrary application data; security considerations will vary by domain of use. Security tools and protocols applicable to text (for example, PGP encryption, checksum validation, password-protected compression) may also be used on RDF documents. Security/privacy protocols must be imposed which reflect the sensitivity of the embedded information.

RDF can express data which is presented to the user, such as RDF Schema labels. Applications rendering strings retrieved from untrusted RDF documents, or using unescaped characters, *SHOULD* use warnings and other appropriate means to limit the possibility that malignant strings might be used to mislead the reader. The security considerations in the media type registration for XML ([RFC3023] section 10) provide additional guidance around the expression of arbitrary data and markup.

RDF uses IRIs as term identifiers. Applications interpreting data expressed in RDF *SHOULD* address the security issues of *Internationalized Resource Identifiers (IRIs)* [RFC3987] Section 8, as well as *Uniform Resource Identifier (URI): Generic Syntax* [RFC3986] Section 7.

Multiple IRIs may have the same appearance. Characters in different scripts may look similar (for instance, a Cyrillic "o" may appear similar to a Latin "o"). A character followed by combining characters may have the same visual representation as another character (for example, LATIN SMALL LETTER "E" followed by COMBINING ACUTE ACCENT has the same visual representation as LATIN SMALL LETTER "E" WITH ACUTE). Any person or application that is writing or interpreting data in RDF must take care to use the IRI that matches the intended semantics, and avoid IRIs that may look similar. Further information about matching visually similar characters can be found in *Unicode Security Considerations* [UNICODE-SECURITY] and *Internationalized Resource Identifiers (IRIs)* [RFC3987] Section 8.

> NOTE
>
> These considerations are a more generic form of Security Considerations for [RDF12-TURTLE], [RDF12-TRIG], [RDF12-N-TRIPLES], and [RDF12-N-QUADS].

§ D. Internationalization Considerations

This section is non-normative.

Unicode [UNICODE] provides a mechanism for signaling direction within a string (see Unicode Bidirectional Algorithm [I18N-Glossary]). RDF provides a mechanism for specifying the base direction of a directional language-tagged string to signal the initial text direction of a string. For most human language strings, but particularly for those whose base direction cannot be accurately determined from the string content, is it valuable to have an external indicator in order to get the proper display and isolation of the value. One example of such an indicator is the [HTML] dir attribute; see [STRING-META].

JSON-LD 1.1 [JSON-LD11] introduced the i18n namespace to use a datatype to specify both the base direction an language tag of an RDF literal.

~~ISSUE 9 (CLOSED)~~: base direction **i18n-tracker** **spec:substantive**

A possible issue for RDF 1.2 is to standardize on a solution for the base direction of strings.

This would possibly include updating to the Abstract Syntax and associated changes to the various Concrete Syntax specifications.

See RDF Literals and Base Directions for possible options.

JSON-LD introduced features for specifying the text direction. These included experimental features compatible with RDF 1.1:
i18n namespace, and rdf:CompoundLiteral.

See the issue for the discussion of further options, and the Working Group page for further discussion.

§ E. IRI Grammar

This section is non-normative.

The following [ABNF] grammar applies the changes from [RFC3987] and [RFC6874] to the section Collected ABNF for URI of [RFC3986] to give a consolidated grammar for IRIs.

This is provided for convenience only. If it differs from definitions in [RFC3986], [RFC3987], or any subsequent updates, then those definitions should be used.

```
IRI           = scheme ":" ihier-part [ "?" iquery ] [ "#" ifragm

ihier-part    = "//" iauthority ipath-abempty
              / ipath-absolute
              / ipath-rootless
              / ipath-empty

IRI-reference = IRI / irelative-ref

absolute-IRI  = scheme ":" ihier-part [ "?" iquery ]

irelative-ref = irelative-part [ "?" iquery ] [ "#" ifragment ]

irelative-part = "//" iauthority ipath-abempty
              / ipath-absolute
              / ipath-noscheme
              / ipath-empty

scheme        = ALPHA *( ALPHA / DIGIT / "+" / "-" / "." )

iauthority    = [ iuserinfo "@" ] ihost [ ":" port ]
iuserinfo     = *( iunreserved / pct-encoded / sub-delims / ":" )
ihost         = IP-literal / IPv4address / ireg-name
port          = *DIGIT

IP-literal    = "[" ( IPv6address / IPv6addrz / IPvFuture  ) "]"

ZoneID        = 1*( unreserved / pct-encoded )

IPv6addrz     = IPv6address "%25" ZoneID

IPvFuture     = "v" 1*HEXDIG "." 1*( unreserved / sub-delims / ":

IPv6address   =                            6( h16 ":" ) ls32
              /                       "::" 5( h16 ":" ) ls32
              / [               h16 ] "::" 4( h16 ":" ) ls32
              / [ *1( h16 ":" ) h16 ] "::" 3( h16 ":" ) ls32
```

```
                / [ *2( h16 ":" ) h16 ] "::" 2( h16 ":" ) ls32
                / [ *3( h16 ":" ) h16 ] "::"    h16 ":"   ls32
                / [ *4( h16 ":" ) h16 ] "::"              ls32
                / [ *5( h16 ":" ) h16 ] "::"              h16
                / [ *6( h16 ":" ) h16 ] "::"

h16             = 1*4HEXDIG
ls32            = ( h16 ":" h16 ) / IPv4address
IPv4address     = dec-octet "." dec-octet "." dec-octet "." dec-oct

dec-octet       = DIGIT                 ; 0-9
                / %x31-39 DIGIT         ; 10-99
                / "1" 2DIGIT            ; 100-199
                / "2" %x30-34 DIGIT     ; 200-249
                / "25" %x30-35          ; 250-255

ireg-name       = *( iunreserved / pct-encoded / sub-delims )

ipath           = ipath-abempty   ; begins with "/" or is empty
                / ipath-absolute  ; begins with "/" but not "//"
                / ipath-noscheme  ; begins with a non-colon segment
                / ipath-rootless  ; begins with a segment
                / ipath-empty     ; zero characters

ipath-abempty  = *( "/" isegment )
ipath-absolute = "/" [ isegment-nz *( "/" isegment ) ]
ipath-noscheme = isegment-nz-nc *( "/" isegment )
ipath-rootless = isegment-nz *( "/" isegment )
ipath-empty    = 0

isegment       = *ipchar
isegment-nz    = 1*ipchar
isegment-nz-nc = 1*( iunreserved / pct-encoded / sub-delims / "@"
               ; non-zero-length segment without any colon ":"

ipchar         = iunreserved / pct-encoded / sub-delims / ":" / "@

iquery         = *( ipchar / iprivate / "/" / "?" )

ifragment      = *( ipchar / "/" / "?" )

iunreserved    = ALPHA / DIGIT / "-" / "." / "_" / "~" / ucschar

ucschar        = %xA0-D7FF / %xF900-FDCF / %xFDF0-FFEF
               / %x10000-1FFFD / %x20000-2FFFD / %x30000-3FFFD
```

```
                    / %x40000-4FFFD / %x50000-5FFFD / %x60000-6FFFD
                    / %x70000-7FFFD / %x80000-8FFFD / %x90000-9FFFD
                    / %xA0000-AFFFD / %xB0000-BFFFD / %xC0000-CFFFD
                    / %xD0000-DFFFD / %xE1000-EFFFD

iprivate        = %xE000-F8FF / %xF0000-FFFFD / %x100000-10FFFD

pct-encoded     = "%" HEXDIG HEXDIG

unreserved      = ALPHA / DIGIT / "-" / "." / "_" / "~"
reserved        = gen-delims / sub-delims
gen-delims      = ":" / "/" / "?" / "#" / "[" / "]" / "@"
sub-delims      = "!" / "$" / "&" / "'" / "(" / ")"
                / "*" / "+" / "," / ";" / "="
```

The ABNF can also be accessed directly from iri-grammar.abnf.

F. Acknowledgments

This section is non-normative.

F.1 Acknowledgments for RDF 1.0

This section is non-normative.

The editors of the original version of the spec were Graham Klyne (Nine by Nine) and Jeremy J. Carroll (Hewlett Packard Labs).

This document contains a significant contribution from Pat Hayes, Sergey Melnik and Patrick Stickler, under whose leadership was developed the framework described in the RDF family of specifications for representing datatyped values, such as integers and dates.

The editors acknowledge valuable contributions from the following: Frank Manola, Pat Hayes, Dan Brickley, Jos de Roo, Dave Beckett, Patrick Stickler, Peter F. Patel-Schneider, Jerome Euzenat, Massimo Marchiori, Tim Berners-Lee, Dave Reynolds and Dan Connolly.

Jeremy Carroll thanks Oreste Signore, his host at the W3C Office in Italy and Istituto di Scienza e Tecnologie dell'Informazione "Alessandro Faedo", part of the Consiglio

Nazionale delle Ricerche, where Jeremy is a visiting researcher.

This document is a product of extended deliberations by the RDFcore Working Group, whose members have included: Art Barstow (W3C), Dave Beckett (ILRT), Dan Brickley (ILRT), Dan Connolly (W3C), Jeremy Carroll (Hewlett Packard), Ron Daniel (Interwoven Inc), Bill dehOra (InterX), Jos De Roo (AGFA), Jan Grant (ILRT), Graham Klyne (Nine by Nine), Frank Manola (MITRE Corporation), Brian McBride (Hewlett Packard), Eric Miller (W3C), Stephen Petschulat (IBM), Patrick Stickler (Nokia), Aaron Swartz (HWG), Mike Dean (BBN Technologies / Verizon), R. V. Guha (Alpiri Inc), Pat Hayes (IHMC), Sergey Melnik (Stanford University) and Martyn Horner (Profium Ltd).

This specification also draws upon an earlier RDF Model and Syntax document edited by Ora Lassilla and Ralph Swick, and RDF Schema edited by Dan Brickley and R. V. Guha. RDF and RDF Schema Working Group members who contributed to this earlier work are: Nick Arnett (Verity), Tim Berners-Lee (W3C), Tim Bray (Textuality), Dan Brickley (ILRT / University of Bristol), Walter Chang (Adobe), Sailesh Chutani (Oracle), Dan Connolly (W3C), Ron Daniel (DATAFUSION), Charles Frankston (Microsoft), Patrick Gannon (CommerceNet), R. V. Guha (Epinions, previously of Netscape Communications), Tom Hill (Apple Computer), Arthur van Hoff (Marimba), Renato Iannella (DSTC), Sandeep Jain (Oracle), Kevin Jones, (InterMind), Emiko Kezuka (Digital Vision Laboratories), Joe Lapp (webMethods Inc.), Ora Lassila (Nokia Research Center), Andrew Layman (Microsoft), Ralph LeVan (OCLC), John McCarthy (Lawrence Berkeley National Laboratory), Chris McConnell (Microsoft), Murray Maloney (Grif), Michael Mealling (Network Solutions), Norbert Mikula (DataChannel), Eric Miller (OCLC), Jim Miller (W3C, emeritus), Frank Olken (Lawrence Berkeley National Laboratory), Jean Paoli (Microsoft), Sri Raghavan (Digital/Compaq), Lisa Rein (webMethods Inc.), Paul Resnick (University of Michigan), Bill Roberts (KnowledgeCite), i Tsuyoshi Sakata (Digital Vision Laboratories), Bob Schloss (IBM), Leon Shklar (Pencom Web Works), David Singer (IBM), Wei (William) Song (SISU), Neel Sundaresan (IBM), Ralph Swick (W3C), Naohiko Uramoto (IBM), Charles Wicksteed (Reuters Ltd.), Misha Wolf (Reuters Ltd.) and Lauren Wood (SoftQuad).

§ F.2 Acknowledgments for RDF 1.1

This section is non-normative.

The editors of the RDF 1.1 version of the spec were Richard Cyganiak (DERI), David Wood (3 Round Stones), and Markus Lanthaler (Graz University of Technology).

The editors acknowledge valuable contributions from Thomas Baker, Tim Berners-Lee, David Booth, Dan Brickley, Gavin Carothers, Jeremy Carroll, Pierre-Antoine Champin, Dan Connolly, John Cowan, Martin J. Dürst, Alex Hall, Steve Harris, Sandro Hawke, Pat Hayes, Ivan Herman, Peter F. Patel-Schneider, Addison Phillips, Eric Prud'hommeaux, Nathan Rixham, Andy Seaborne, Leif Halvard Silli, Guus Schreiber, Dominik Tomaszuk, and Antoine Zimmermann.

The membership of the RDF Working Group included Thomas Baker, Scott Bauer, Dan Brickley, Gavin Carothers, Pierre-Antoine Champin, Olivier Corby, Richard Cyganiak, Souripriya Das, Ian Davis, Lee Feigenbaum, Fabien Gandon, Charles Greer, Alex Hall, Steve Harris, Sandro Hawke, Pat Hayes, Ivan Herman, Nicholas Humfrey, Kingsley Idehen, Gregg Kellogg, Markus Lanthaler, Arnaud Le Hors, Peter F. Patel-Schneider, Eric Prud'hommeaux, Yves Raimond, Nathan Rixham, Guus Schreiber, Andy Seaborne, Manu Sporny, Thomas Steiner, Ted Thibodeau, Mischa Tuffield, William Waites, Jan Wielemaker, David Wood, Zhe Wu, and Antoine Zimmermann.

§ F.3 Acknowledgments for RDF 1.2

This section is non-normative.

In addition to the editors, the following people have contributed to this specification: Dominik Tomaszuk, Peter F. Patel-Schneider, Sarven Capadisli, Ted Thibodeau Jr, and Thomas Tanon

Members of the RDF-star Working Group Group included Vladimir Alexiev, Amin Anjomshoaa, Julián Arenas-Guerrero, Dörthe Arndt, Bilal Ben Mahria, Erich Bremer, Kurt Cagle, Rémi Ceres, Pierre-Antoine Champin, Souripriya Das, Daniil Dobriy, Enrico Franconi, Jeffrey Phillips Freeman, Fabien Gandon, Benjamin Goering, Adrian Gschwend, Olaf Hartig, Timothée Haudebourg, Ian Horrocks, Gregg Kellogg, Mark Kim, Jose Emilio Labra Gayo, Ora Lassila, Richard Lea, Niklas Lindström, Pasquale Lisena, Thomas Lörtsch, Matthew Nguyen, Peter Patel-Schneider, Thomas Pellissier Tanon, Dave Raggett, Jean-Yves ROSSI, Felix Sasaki, Andy Seaborne, Ruben Taelman, Ted Thibodeau Jr, Dominik Tomaszuk, Raphaël Troncy, William Van Woensel, Gregory Williams, Jesse Wright, Achille Zappa, and Antoine Zimmermann.

> EDITOR'S NOTE
>
> Recognize members of the Task Force? Not an easy to find list of contributors.

§ G. Changes between RDF 1.1 and RDF 1.2

This section is non-normative.

- Added 4.3 Dataset as a Set of Quads for informative definition of a quad.
- Added 1.5 Triple Terms and Reification and definitions for triple term and asserted triple and extended the definition of RDF triple to permit triple terms as objects. Also defines reifier and reifying triple.
- Added the base direction element as part of a literal, and a description of its use in 3.6 Initial Text Direction.
- Improved the use of IRI terminology, and added E. IRI Grammar. This improves the language using relative IRI references and clarifies that, in the abstract syntax, IRIs are resolved, avoiding the incorrect use of "absolute IRI".
- Changed reference from DOM4, which was not a recommendation at the time, to [DOM], making the definitions of `rdf:HTML` and `rdf:XMLLiteral` datatypes normative.
- Added A. Additional Datatypes and moved the sections about the `rdf:HTML` and `rdf:XMLLiteral` datatypes to this appendix.
- Added the `rdf:JSON` datatype, the definition of which is adopted from Section 10.2 The `rdf:JSON` Datatype in [JSON-LD11]. Note that the value space defined here updates the value space of the `rdf:JSON` datatype defined in *JSON-LD 1.1* [JSON-LD11]
- Clarify Unicode terminology, using Unicode code points, and restriction to the XML Char production. Also removes obsolete recommendations for the use of Normalization Form C in literals. Adds a definition of string that can be used in other RDF documents.
- Minor edit to improve the example about distinguishing literals, IRIs, and blank nodes in 3.1 Triples.
- Implementations were previously allowed to normalize language tags to lower case, which made it ambiguous whether two literals with language tags that differed only by case represented the same literal, or distinct literals. RDF 1.2 requires that language tags be case-insensitively unique but does not specify the common formatting to be used. Two literals with the same lexical form and language tags that differ only by case are the same literal. Implementations can either follow the advice to normalize to lower

case, use the recommended BCP47 format, or do something else, as long it is performed consistently.

- Removed the section on the canonical mapping for the `rdf:XMLLiteral` datatype.
- Refer to the definition and discussion of RDF Semantics, "recognizing" datatype IRIs, instead of *Recognized datatype IRIs*.
- The informal terminolgy "RDF processor" has been removed.

> NOTE
>
> A detailed overview of the differences between RDF versions 1.1 and 1.2 can be found in *What's New in RDF 1.2* [RDF12-NEW].

H. Index

H.1 Terms defined by this specification

- fragment identifiers §6.
- Full conformance §2.
- generalized RDF dataset §7.
- generalized RDF graph §7.
- generalized RDF triple §7.
- graph name §4.
- Inconsistency §1.8
- IRI §3.2
- IRI equality §3.2
- IRI references §3.2
- isomorphic §3.8
- language tag §3.3
- language-tagged string §3.3
- lexical form §3.3
- lexical space §5.
- lexical-to-value mapping §5.
- literal §3.3
- Literal term equality §3.3
- literal value §3.3
- logical expression §1.8
- named graphs §4.
- namespace §1.4
- namespace IRI §1.4
- namespace prefix §1.4
- nodes §3.1
- object §3.5
- predicate §3.5
- property §1.2
- proposition §1.2
- quad §4.3
- RDF dataset §4.
- RDF document §1.9
- RDF graph §3.

- RDF source §1.6
- RDF statement §1.2
- RDF terms §3.1
- RDF triple §3.1
- RDF vocabulary §1.4
- RDF-compatible XSD types §5.1
- `rdf:HTML` §A.1
- `rdf:JSON` §A.3
- `rdf:XMLLiteral` §A.2
- referent §1.3
- reifier §1.5
- reifying triple §1.5
- relative IRI references §3.2
- resources §1.2
- simple literals §3.3
- Skolem IRI §3.7
- string §2.1
- subject §3.5
- symmetric RDF dataset §7.
- symmetric RDF graph §7.
- symmetric RDF triple §7.
- triple term §3.5
- URIs §3.2
- value space §5.

§ H.2 Terms defined by reference

- [BCP47] defines the following:
  - BCP 47 section 4.5
  - section 2.2.9
- [DOM] defines the following:
  - DocumentFragment

    - DOM nodes
    - `isEqualNode(otherNode)` (for `Node`)
    - `normalize()` (for `Node`)
- [HTML] defines the following:
    - dir attribute
- [HTML5] defines the following:
    - HTML fragment parsing algorithm
- [I18N-GLOSSARY] defines the following:
    - bidi isolation
    - case sensitive matching
    - code units
    - Normalization Form C
    - spillover effects
    - surrogate code points
    - Unicode Bidirectional Algorithm
    - Unicode character encoding
    - Unicode code points
    - Unicode scalar values
- [INFRA] defines the following:
    - items
    - key
    - lists
    - map entries
    - maps
    - null
    - size
    - size
    - true, false
    - value
- [JSON-LD11] defines the following:
    - i18n namespace
    - Section 10.2 The rdf:JSON Datatype
- [OWL2-OVERVIEW] defines the following:
    - OWL 2
- [OWL2-SYNTAX] defines the following:

    - custom datatypes
- [RDF12-SEMANTICS] defines the following:
    - entailment regime
    - RDF Semantics, "recognizing"
    - semantic extensions
- [RDF12-XML] defines the following:
    - @parseType="literal"
- [RFC3986] defines the following:
    - base IRI can be established
    - Collected ABNF for URI
    - defined in RFC 3986
    - resolved
    - resolved against
- [RFC3987] defines the following:
    - section 1.3
    - section 3.1
    - Section 5
    - section 5.3.1
- [RFC8259] defines the following:
    - JSON Array
    - JSON Grammar
    - JSON literal name
    - JSON Number
    - JSON Object
    - JSON String
- [SWBP-N-ARYRELATIONS] defines the following:
    - indirectly expressed in RDF
- [SWBP-XSCH-DATATYPES] defines the following:
    - user-defined simple XML Schema datatypes
- [WEBARCH] defines the following:
    - content negotiation
    - dereferences
    - IRI collision
    - IRI owner
    - URI persistence

- [XML11] defines the following:
  - Char
  - XML content
- [XMLSCHEMA11-2] defines the following:
  - doubleLexicalMap
  - floatPtRound
  - xsd:anyURI
  - xsd:base64Binary
  - xsd:boolean
  - xsd:byte
  - xsd:date
  - xsd:dateTime
  - xsd:dateTimeStamp
  - xsd:dayTimeDuration
  - xsd:decimal
  - xsd:double
  - xsd:duration
  - xsd:ENTITIES
  - xsd:ENTITY
  - xsd:float
  - xsd:gDay
  - xsd:gMonth
  - xsd:gMonthDay
  - xsd:gYear
  - xsd:gYearMonth
  - xsd:hexBinary
  - xsd:ID
  - xsd:IDREF
  - xsd:IDREFS
  - xsd:int
  - xsd:integer
  - xsd:language
  - xsd:long
  - xsd:Name
  - xsd:NCName

- xsd:negativeInteger
- xsd:NMTOKEN
- xsd:NMTOKENS
- xsd:nonNegativeInteger
- xsd:nonPositiveInteger
- xsd:normalizedString
- xsd:NOTATION
- xsd:positiveInteger
- xsd:QName
- xsd:short
- xsd:string
- xsd:time
- xsd:token
- xsd:unsignedByte
- xsd:unsignedInt
- xsd:unsignedLong
- xsd:unsignedShort
- xsd:yearMonthDuration

§ I. Issue summary

Issue 9: base direction

§ J. References

§ J.1 Normative references

[BCP47]

Tags for Identifying Languages. A. Phillips, Ed.; M. Davis, Ed. IETF. September 2009. Best Current Practice. URL: https://www.rfc-editor.org/rfc/rfc5646

[DOM]

DOM Standard. Anne van Kesteren. WHATWG. Living Standard. URL: https://dom.spec.whatwg.org/

[HTML5]

HTML5. Ian Hickson; Robin Berjon; Steve Faulkner; Travis Leithead; Erika Doyle Navara; Theresa O'Connor; Silvia Pfeiffer. W3C. 27 March 2018. W3C Recommendation. URL: https://www.w3.org/TR/html5/

[I18N-GLOSSARY]

Internationalization Glossary. Richard Ishida; Addison Phillips. W3C. 17 October 2024. W3C Working Group Note. URL: https://www.w3.org/TR/i18n-glossary/

[INFRA]

Infra Standard. Anne van Kesteren; Domenic Denicola. WHATWG. Living Standard. URL: https://infra.spec.whatwg.org/

[RDF11-TESTCASES]

RDF 1.1 Test Cases. Gregg Kellogg; Markus Lanthaler. W3C. 25 February 2014. W3C Working Group Note. URL: https://www.w3.org/TR/rdf11-testcases/

[RDF12-N-QUADS]

RDF 1.2 N-Quads. Gregg Kellogg; Dominik Tomaszuk. W3C. 19 December 2024. W3C Working Draft. URL: https://www.w3.org/TR/rdf12-n-quads/

[RDF12-N-TRIPLES]

RDF 1.2 N-Triples. Gregg Kellogg; Dominik Tomaszuk. W3C. 19 December 2024. W3C Working Draft. URL: https://www.w3.org/TR/rdf12-n-triples/

[RDF12-TRIG]

RDF 1.2 TriG. Gregg Kellogg; Dominik Tomaszuk. W3C. 19 December 2024. W3C Working Draft. URL: https://www.w3.org/TR/rdf12-trig/

[RDF12-TURTLE]

RDF 1.2 Turtle. Gregg Kellogg; Dominik Tomaszuk. W3C. 9 January 2025. W3C Working Draft. URL: https://www.w3.org/TR/rdf12-turtle/

[RFC2119]

Key words for use in RFCs to Indicate Requirement Levels. S. Bradner. IETF. March 1997. Best Current Practice. URL: https://www.rfc-editor.org/rfc/rfc2119

[RFC3629]

UTF-8, a transformation format of ISO 10646. F. Yergeau. IETF. November 2003. Internet Standard. URL: https://www.rfc-editor.org/rfc/rfc3629

[RFC3986]

Uniform Resource Identifier (URI): Generic Syntax. T. Berners-Lee; R. Fielding; L. Masinter. IETF. January 2005. Internet Standard. URL: https://www.rfc-editor.org/rfc/rfc3986

[RFC3987]

Internationalized Resource Identifiers (IRIs). M. Duerst; M. Suignard. IETF. January 2005. Proposed Standard. URL: https://www.rfc-editor.org/rfc/rfc3987

[RFC7493]

The I-JSON Message Format. T. Bray, Ed. IETF. March 2015. Proposed Standard. URL: https://www.rfc-editor.org/rfc/rfc7493

[RFC8174]

Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words. B. Leiba. IETF. May 2017. Best Current Practice. URL: https://www.rfc-editor.org/rfc/rfc8174

[RFC8259]

The JavaScript Object Notation (JSON) Data Interchange Format. T. Bray, Ed. IETF. December 2017. Internet Standard. URL: https://www.rfc-editor.org/rfc/rfc8259

[RFC8615]

Well-Known Uniform Resource Identifiers (URIs). M. Nottingham. IETF. May 2019. Proposed Standard. URL: https://www.rfc-editor.org/rfc/rfc8615

[Unicode]

The Unicode Standard. Unicode Consortium. URL: https://www.unicode.org/versions/latest/

[XML-NAMES]

Namespaces in XML 1.0 (Third Edition). Tim Bray; Dave Hollander; Andrew Layman; Richard Tobin; Henry Thompson et al. W3C. 8 December 2009. W3C Recommendation. URL: https://www.w3.org/TR/xml-names/

[XML11]

Extensible Markup Language (XML) 1.1 (Second Edition). Tim Bray; Jean Paoli; Michael Sperberg-McQueen; Eve Maler; François Yergeau; John Cowan et al. W3C. 16 August 2006. W3C Recommendation. URL: https://www.w3.org/TR/xml11/

[XMLSCHEMA11-2]

W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes. David Peterson; Sandy Gao; Ashok Malhotra; Michael Sperberg-McQueen; Henry Thompson; Paul V. Biron et al. W3C. 5 April 2012. W3C Recommendation. URL: https://www.w3.org/TR/xmlschema11-2/

§ J.2 Informative references

[ABNF]

Augmented BNF for Syntax Specifications: ABNF. D. Crocker, Ed.; P. Overell. IETF. January 2008. Internet Standard. URL: https://www.rfc-editor.org/rfc/rfc5234

[COOLURIS]

Cool URIs for the Semantic Web. Leo Sauermann; Richard Cyganiak. W3C. 3 December 2008. W3C Working Group Note. URL: https://www.w3.org/TR/cooluris/

[HTML]

HTML Standard. Anne van Kesteren; Domenic Denicola; Dominic Farolino; Ian Hickson; Philip Jägenstedt; Simon Pieters. WHATWG. Living Standard. URL: https://html.spec.whatwg.org/multipage/

[HTML-RDFA]

HTML+RDFa 1.1 - Second Edition. Manu Sporny. W3C. 17 March 2015. W3C Recommendation. URL: https://www.w3.org/TR/html-rdfa/

[JSON-LD11]

JSON-LD 1.1. Gregg Kellogg; Pierre-Antoine Champin; Dave Longley. W3C. 16 July 2020. W3C Recommendation. URL: https://www.w3.org/TR/json-ld11/

[LINKED-DATA]

Linked Data Design Issues. Tim Berners-Lee. W3C. 27 July 2006. W3C-Internal Document. URL: https://www.w3.org/DesignIssues/LinkedData.html

[OWL2-OVERVIEW]

OWL 2 Web Ontology Language Document Overview (Second Edition). W3C OWL Working Group. W3C. 11 December 2012. W3C Recommendation. URL: https://www.w3.org/TR/owl2-overview/

[OWL2-SYNTAX]

OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition). Boris Motik; Peter Patel-Schneider; Bijan Parsia. W3C. 11 December 2012. W3C Recommendation. URL: https://www.w3.org/TR/owl2-syntax/

[RDF-CONCEPTS-20040210]

Resource Description Framework (RDF): Concepts and Abstract Syntax. Graham Klyne; Jeremy Carroll. W3C. 10 February 2004. W3C Recommendation. URL: https://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

[RDF11-CONCEPTS]

RDF 1.1 Concepts and Abstract Syntax. Richard Cyganiak; David Wood; Markus Lanthaler. W3C. 25 February 2014. W3C Recommendation. URL: https://www.w3.org/TR/rdf11-concepts/

[RDF11-DATASETS]

RDF 1.1: On Semantics of RDF Datasets. Antoine Zimmermann. W3C. 25 February 2014. W3C Working Group Note. URL: https://www.w3.org/TR/rdf11-datasets/

[RDF12-NEW]

What's New in RDF 1.2. David Wood. W3C. DNOTE. URL: https://w3c.github.io/rdf-new/spec/

[RDF12-PRIMER]

RDF 1.2 Primer. Guus Schreiber; Yves Raimond. W3C. DNOTE. URL: https://w3c.github.io/rdf-primer/spec/

[RDF12-SCHEMA]

RDF 1.2 Schema. Dominik Tomaszuk; Timothée Haudebourg. W3C. 19 December 2024. W3C Working Draft. URL: https://www.w3.org/TR/rdf12-schema/

[RDF12-SEMANTICS]

RDF 1.2 Semantics. Peter Patel-Schneider; Dörthe Arndt; Timothée Haudebourg. W3C. 19 December 2024. W3C Working Draft. URL: https://www.w3.org/TR/rdf12-semantics/

[RDF12-XML]

RDF 1.2 XML Syntax. Gregg Kellogg. W3C. 19 December 2024. W3C Working Draft. URL: https://www.w3.org/TR/rdf12-xml/

[RDFA-CORE]

RDFa Core 1.1 - Third Edition. Ben Adida; Mark Birbeck; Shane McCarron; Ivan Herman et al. W3C. 17 March 2015. W3C Recommendation. URL: https://www.w3.org/TR/rdfa-core/

[RFC3023]

XML Media Types. M. Murata; S. St. Laurent; D. Kohn. IETF. January 2001. Proposed Standard. URL: https://www.rfc-editor.org/rfc/rfc3023

[RFC3492]

Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA). A. Costello. IETF. March 2003. Proposed Standard. URL: https://www.rfc-editor.org/rfc/rfc3492

[RFC6874]

Representing IPv6 Zone Identifiers in Address Literals and Uniform Resource Identifiers. B. Carpenter; S. Cheshire; R. Hinden. IETF. February 2013. Proposed Standard. URL: https://www.rfc-editor.org/rfc/rfc6874

[SPARQL11-QUERY]

SPARQL 1.1 Query Language. Steven Harris; Andy Seaborne. W3C. 21 March 2013. W3C Recommendation. URL: https://www.w3.org/TR/sparql11-query/

[SPARQL12-CONCEPTS]

SPARQL 1.2 Concepts. The W3C RDF-star Working Group. W3C. W3C Working Draft. URL: https://w3c.github.io/sparql-concepts/spec/

[SPARQL12-ENTAILMENT]

SPARQL 1.2 Entailment Regimes. Peter Patel-Schneider. W3C. 19 December 2024.

W3C Working Draft. URL: https://www.w3.org/TR/sparql12-entailment/

[SPARQL12-FEDERATED-QUERY]

SPARQL 1.2 Federated Query. Ruben Taelman; Gregory Williams. W3C. 19 December 2024. W3C Working Draft. URL: https://www.w3.org/TR/sparql12-federated-query/

[SPARQL12-GRAPH-STORE-PROTOCOL]

SPARQL 1.2 Graph Store Protocol. Andy Seaborne; Thomas Pellissier Tanon. W3C. 19 December 2024. W3C Working Draft. URL: https://www.w3.org/TR/sparql12-graph-store-protocol/

[SPARQL12-NEW]

What's New in SPARQL 1.2. The W3C RDF-star Working Group. W3C. W3C Working Draft. URL: https://w3c.github.io/sparql-new/spec/

[SPARQL12-PROTOCOL]

SPARQL 1.2 Protocol. Andy Seaborne; Ruben Taelman; Gregory Williams; Thomas Pellissier Tanon. W3C. 19 December 2024. W3C Working Draft. URL: https://www.w3.org/TR/sparql12-protocol/

[SPARQL12-QUERY]

SPARQL 1.2 Query Language. Olaf Hartig; Andy Seaborne; Ruben Taelman; Gregory Williams; Thomas Pellissier Tanon. W3C. 27 December 2024. W3C Working Draft. URL: https://www.w3.org/TR/sparql12-query/

[SPARQL12-RESULTS-CSV-TSV]

SPARQL 1.2 Query Results CSV and TSV Formats. Ruben Taelman; Gregory Williams; Thomas Pellissier Tanon. W3C. 19 December 2024. W3C Working Draft. URL: https://www.w3.org/TR/sparql12-results-csv-tsv/

[SPARQL12-RESULTS-JSON]

SPARQL 1.2 Query Results JSON Format. Andy Seaborne; Ruben Taelman; Gregory Williams; Thomas Pellissier Tanon. W3C. 19 December 2024. W3C Working Draft. URL: https://www.w3.org/TR/sparql12-results-json/

[SPARQL12-RESULTS-XML]

SPARQL 1.2 Query Results XML Format. Ruben Taelman; Dominik Tomaszuk; Thomas Pellissier Tanon. W3C. 27 December 2024. W3C Working Draft. URL: https://www.w3.org/TR/sparql12-results-xml/

[SPARQL12-SERVICE-DESCRIPTION]

SPARQL 1.2 Service Description. Ruben Taelman; Gregory Williams. W3C. 19 December 2024. W3C Working Draft. URL: https://www.w3.org/TR/sparql12-service-description/

[SPARQL12-UPDATE]

SPARQL 1.2 Update. Ruben Taelman; Andy Seaborne; Thomas Pellissier Tanon. W3C. 19 December 2024. W3C Working Draft. URL: https://www.w3.org/TR/sparql12-update/

[STRING-META]

Strings on the Web: Language and Direction Metadata. Richard Ishida; Addison Phillips. W3C. 17 October 2024. W3C Working Group Note. URL: https://www.w3.org/TR/string-meta/

[SWBP-N-ARYRELATIONS]

Defining N-ary Relations on the Semantic Web. Natasha Noy; Alan Rector. W3C. 12 April 2006. W3C Working Group Note. URL: https://www.w3.org/TR/swbp-n-aryRelations/

[SWBP-XSCH-DATATYPES]

XML Schema Datatypes in RDF and OWL. Jeremy Carroll; Jeff Pan. W3C. 14 March 2006. W3C Working Group Note. URL: https://www.w3.org/TR/swbp-xsch-datatypes/

[UNICODE-SECURITY]

Unicode Security Considerations. Mark Davis; Michel Suignard. Unicode Consortium. 19 September 2014. Unicode Technical Report #36. URL: https://www.unicode.org/reports/tr36/tr36-15.html

[URL]

URL Standard. Anne van Kesteren. WHATWG. Living Standard. URL: https://url.spec.whatwg.org/

[VOCAB-ORG]

The Organization Ontology. Dave Reynolds. W3C. 16 January 2014. W3C Recommendation. URL: https://www.w3.org/TR/vocab-org/

[WEBARCH]

Architecture of the World Wide Web, Volume One. Ian Jacobs; Norman Walsh. W3C. 15 December 2004. W3C Recommendation. URL: https://www.w3.org/TR/webarch/